# MODELNET:
# A MODELING AND SIMULATION ENVIRONMENT FOR LEGACY PROGRAMS

Dr. John H. Schaibly
Science Applications International Corporation
10260 Campus Point Drive
San Diego, CA 92121
and
John E. Camp
Air Force Research Laboratory / IFSD
2241 Avionics Circle
WPAFB, Ohio 45433

19981029 109

## Abstract

MODELNET is a modeling and simulation environment for running legacy codes on UNIX platforms. It
provides a user shell to organize and run a set of related models and exchange data between them. MODELNET assists the model *user* by providing GUI input, visual model linking tools, context sensitive help, integrated documentation, and shared I/O utilities. It assists the model *developer* by organizing model directories, compilation tools, and shared libraries. It supports collaborative engineering environments where model and data sharing are important. While MODELNET components are available at essentially no cost, Science Applications International Corporation (SAIC) offers model integration services and integration training.

## Background

Over many years engineering software models have been developed by industry, government and university with varying degrees of quality and utility. We are all familiar with programs that are superbly documented, have a handsome interface, but whose technical capabilities are limited or give wrong answers, and other models which are poorly documented, riddled with untrapped errors, clumsy inputs, but which give highly accurate answers, and, with hundreds or thousands of users, are considered standards.

Each model has a history from concept through marketing of that concept, prototype development, development, delivery and maintenance and user support. Generally, the models which "rise to the top" and are still in widespread use after many years are technically accurate and solve useful, persistent problems. But there are many models abandoned along the way not because of deficiency of quality, but for political, organizational or personal reasons. If a government program is canceled, if a key developer retires, or if a developing company fails, good technical software and technical information disappears or is filed away out of view. Ironically, after a few years have

DTIC QUALITY INSPECTED 4

passed, there is often a new requirement in a new program office who funds a new contract to a new contractor to rediscover and redevelop the old technology. Developers who have spent a lifetime in this environment are often painfully aware of how few of their products were widely used and valued. Others take advantage of the quick obsolescence to resell the same technology repeatedly to different customers causing inflated costs and wasted effort.

Ignoring the political, organizational and personal aspects of software obsolescence, there are technical reasons why legacy software ceases to be used. As computer technology progresses, users are reluctant to use programs that seem outmoded. Younger users familiar and trained on software developed in the Age of Microsoft will be impatient and frustrated at older methods of input based on the 80 column punched card or even the menu driven inputs of the 1970's and 80's. They are not comfortable with poorly defined or complicated input instructions. MODELNET addresses these problems.

Where there used to be teams of analysts working on a problem, now fewer, less experienced users are expected to get answers often without the expert advice and oversight present in the past. MODELNET also addresses this problem.

Then there is the multiplicity of models. If more than one legacy model is available for solving a particular problem, the user must choose between them based on the usual speed/accuracy tradeoff. How does the accuracy of model results affect the results of subsequent model calculations using those results as inputs? MODELNET addresses these problems.

Expanding problem domains may require several models to be linked in sequence. Thus data passing between models becomes a real issue. Much effort is spent interfacing data between models. This becomes a difficult and error-prone process when the data source and destination work in differing units, resolutions, or datatype. Often inputs require calculating a combination of outputs from one or more models requiring sequential control over the models. MODELNET addresses these problems.

In modern software engineering, there is an emphasis on model reuse, distributed processing and model decomposition. In some cases, a set of existing sub calculations need to be integrated into a single aggregate model. In other cases, an existing complex calculation requires breaking up into a set of components which may further be required to be reassembled in a new configuration. This aggregation / disaggregation may occur at several levels: procedural level, subroutine level, or in modern software, object level. MODELNET addresses these issues.

Finally, MODELNET supports the collaborative engineering environment. As models are developed by different developers at different locations, even using different platforms, they can share their work easily. The MODELNET development process has sufficient documented standards that guarantees portability to other MODELNET systems running on other UNIX platforms.

MODELNET is not the first attempt at a modeling and simulation environment / system to address these various issues Some of these have come and gone and others are on hold awaiting funding. MODELNET is different in that the framework is made up of components which are either public domain or ubiquitous off the shelf existing products which have life independent of MODELNET; the system can grow and evolve as the components grow.

## MODELNET components

MODELNET is a development and user framework; a block diagram is shown in Figure 1. It consists of Khoros 2.2, Browser, CLIPS6 (C Language Integrated Production System) Expert System tool from NASA, and a library of Freeware utilities. By using only freely available components, MODELNET is basically free. By using components which are common off-the-shelf and supported by many thousands of users, MODELNET works today, is maintained by the component developer, and will leverage advancements of these components to advancements in MODELNET. Furthermore, users will see familiar user interfaces and not have to learn yet another one. What users are not familiar with Netscape Browsers, spreadsheet programs, and point and click GUI's? MODELNET uses the components unmodified so that, as new backward compatible versions appear, all implemented models will be compatible.

## Khoros 2.2 component in MODELNET

Khoros 2.2 is a major part of MODELNET. It is developed by Khoral Research Inc. in Albuquerque, NM as an image processing environment. It has a graphical programming user interface called Cantata which allows icons ("glyphs") to be moved around the screen using the mouse, much like arranging boxes using a graphics draw program. Each glyph represents a program process. There are handles on the glyph which can be connected to handles from other glyphs indicating that data will flow between them. Figure 2 shows a Cantata screen containing five glyphs representing, from left to right, an input scene image file as generated by GTRENDER, a noise addition process, a jitter process which uses a vibration power density function, and an output viewer. This set of five glyphs is referred to as a "glyph (or model) network". When the "RUN" button is pressed, Cantata starts the first process and when it is finished, the second process starts automatically. Data is passed between the glyphs only when calculation of that data is complete.

In addition to a large library of image processing and image data handling processes included with Cantata, control glyphs can be embedded in the network. For example, there can be "if - then - else" branches, where completion of a process can trigger one of two succeeding processes depending on some user specified or other process specified condition. Repetition loops can be added to repeat sequentially running a network systematically varying user specified parameters. The network has intelligence so that in multiple runs it will not repeat running processes if the inputs have not changed.

In Khoros, integrated program modules are called "objects" (not to be confused with "object oriented"). Each object contains an executable, source code, supporting code for data manipulation, GUI data and help files. One of the Khoros development modules (Composer) sets up development directories, make files, and writes code shells for integration software. Figure 3 shows the modules of Khoros which we use in MODELNET. Included in Composer is a GUI builder and an implicit GUI visual structure which is the same for all integrated models. A set of objects is called a Toolbox which is managed by another Khoros development module called Craftsman. This allows groups of models for specific application areas to be collectively turned on or off, or delivered to different customers, or shared by an authorized community of users.

While Khoros was developed for integrating image processing applications, it has the facility for integrating much more complex applications. The Khoros training courses (which are very good) only deal with software development of simple models and do not address issues arising from complex model integration. SAIC has developed a methodology for integrating such complex models for two toolboxes described below, and the methodology appears to be applicable to a large class of application areas.

## Browser component in MODELNET

The browser component brings to MODELNET the capability of hypertext documentation and context sensitive help systems. We have used Netscape, although Mosaic or HotJava browsers work similarly. We have linked Khoros objects and Netscape so that they can call each other. Thus, if a user clicks on the help button of a glyph's GUI, the help page is brought up in the browser window. All the hypertext features of the browser are brought to bear on providing the level of help required. Links to the complete documentation system are possible making a truly integrated system. The documentation can be in HTML accessible by the browser directly, and in PDF, also accessible by most browsers. Since these formats are readily editable, the user can tailor documentation and hypertext to his own needs. Figure 4 shows the uses of the Browser in MODELNET.

If the platform is connected to the internet, all those features are available to the MODELNET system. For example, sources for many kinds of data are then available to MODELNET users. E-mail contact to model developers (presumably for a small fee) is available on the same platform. The browser also provides a client-side interface to other servers for distributed computing applications.

## CLIPS component in MODELNET

Figure 5 shows the uses of CLIPS 6.0 in MODELNET. Prototypes for CLIPS integration have been produced, but no delivered examples have been implemented to date. The design for CLIPS integration includes checking input variables, output variables (for obvious errors), and most importantly, intermodel consistency of inputs and outputs to assure compatible units and ranges of validity. The rule database is also be editable by the user.

## Multiple Levels of Model Aggregation

MODELNET supports multiple levels of model aggregation depending on user need. Typically, an experienced user must set up a network of model components validating all input variables, and setting up the required databases. However, using a Khoros feature called encapsulation, the set of data and models can be combined into one process with a reduced set of inputs (typically scenario definition variables such as range, altitude. The encapsulated model can then be run easily by a relatively unsophisticated user who is protected from worrying about unfamiliar inputs and processes.

MODELNET also facilitates disaggregation. For example, many engineering codes already contain calls to subprocesses. These are very easy to separate in MODELNET, allowing substitution of alternative models for one or more of the subprocesses. Of course, data translation of inputs and outputs may be required. There is also a trend in the M&S community to componentize existing legacy models, that is, functionally decompose the models into components which are often rewritten as object oriented code. We are also looking at adapting MODELNET to these types of modules as well, but we have not yet demonstrated that capability.

## Current MODELNET applications

SAIC has integrated two major toolboxes using MODELNET. One, the Advanced Electromagnetic Model for Aerial Targeting (AEM*AT), developed for AFRL, integrated a set of IR signature and sensor codes for aircraft. Models integrated included SPIRITS 4.2, an IR aircraft signature model; EMBED, a target embedding model for combining target and textured background scenes; LASERX, an active signature model using solids models; EOSSIM, a sensor model, and XPATCH version 1, a radar cross section code. A preliminary integrated version of Georgia Tech's scene generation programs GTRENDER and GTSIG is also included. This simulation can be run in an animation, moving the target and sensor according to a detailed flyout trajectory.

The second toolbox developed by SAIC was NEOTAM, the NATO Electro-Optical Target Model specializing in missile plume signature models developed for the Research

Study Group RSG-18. Models from the participating NATO countries were integrated to provide a common model set for predicting missile signatures.

## Evolutionary Integration into MODELNET

The integration procedure SAIC has developed for MODELNET is a multi-step process which allows basic integration with very little effort, and increasing utility with increasing integration. This accommodates customers who have limited resources or who want frequent review and control over the final product. The process may be broken into the following levels:

Level 1 - porting the legacy model to the desired platform and verifying its operation. This can be done either by the integrator or by the customer.

Level 2 - calling the model from the Khoros Cantata workspace as a standalone process; use legacy input method / files and data.

Level 3 - developing a Quick GUI for changing frequent inputs such as scenario variables, or inputs depending on other models already integrated into MODELNET, or input variables for which multiple values are routinely run (batch runs). The user would enter all inputs using the legacy input methodology, and only the variables in the Quick GUI would be changed at network runtime. Quick GUIs consist of only one page of point and click inputs. This level includes writing the software for model data integration.

Level 4 - developing a comprehensive GUI for most of the inputs for the model. This can result in many (tens of) pages of inputs which are organized according to the Khoros GUI paradigm. SAIC has extended the Khoros capability of using either Athena or Motif widgets to include Java based widgets as well. User documentation is also rewritten to accommodate the new GUI inputs.

Level 5 - completing documentation and expert system. This final level will best be completed after the familiar user has been working with the MODELNET implementation of the model and can write the requirements. Hypertext online documentation is provided and integrated with other documentation in the toolbox. The expert system rules are defined by the user and formatted by the developer to assist the inexperienced user and to help all users avoid and correct mistakes.

Levels 1 and 2 integration typically take only a few days. Level 3 requires development of software to parse and write the legacy input file formats. The other levels of integration require correspondingly more effort.

Model and toolbox documentation are done both in HTML and Adobe Acrobat (PDF) formats. Both of these formats are commonplace and allow the user to modify the documentation for his own purposes. Documentation forms can include written reports, color graphics, AVI or QuickTime movies, sound, and internet sources as well as e-mail.

## MODELNET Portability

MODELNET applications are developed on either (or both) the Sun or Silicon Graphics workstations. When Khoros becomes available on the NT, our toolboxes can be ported easily to that platform as well. Portability of MODELNET depends on portability of Khoros and the Browser. Application toolboxes developed on one platform can typically be ported to one of the other supported platforms in a day or two. Supported platforms include:

| Platform | Khoros | Netscape | CLIPS |
|---|---|---|---|
| Sun ( > Solaris 2.4, > SunOS 5.4) | x | x | x |
| SGI ( > IRIX 5.2) | x | x | x |
| LINUX(PC) (Yggdrasil, Redhat) | x | x | x |
| FreeBSD, BSD/05 (PC) | x | x | x |
| IBM RS/6000 | x | x | x |
| HP 9000.( > 700) | x | x | x |
| DEC Alpha AXP (OSF/1 > 3.0) | x | x | x |
| DG Aviion (DG/UX) | x | x | x |
| Macintosh | | x | x |
| DOS | | x | x |
| Windows 95 / NT | | x | |

## MODELNET Availability

MODELNET is a software integration service sold by SAIC to military and industrial customers. Customers who have software which they would like to integrate under this system should contract with the MODELNET development team for all or some of the integration effort. SAIC also encourages teaming arrangements with similar companies on other software integration / development procurements. If desired, SAIC will provide tutorial instruction and continuing user support on all phases of the integration process allowing customers to perform their own integration in the future. If integrated models are of interest to a wider community of users and the customer desires to market them commercially, SAIC can work with the customer to provide a distribution package. The customer will retain possession and distribution rights to any legacy models unless special agreements are made.

Interested parties please contact Dr. John Schaibly at SAIC (john.h.schaibly@cpmx.saic.com) or call him at (619)646-4025 for information on the MODELNET system.

Figure1 - MODELNET Block Diagram.

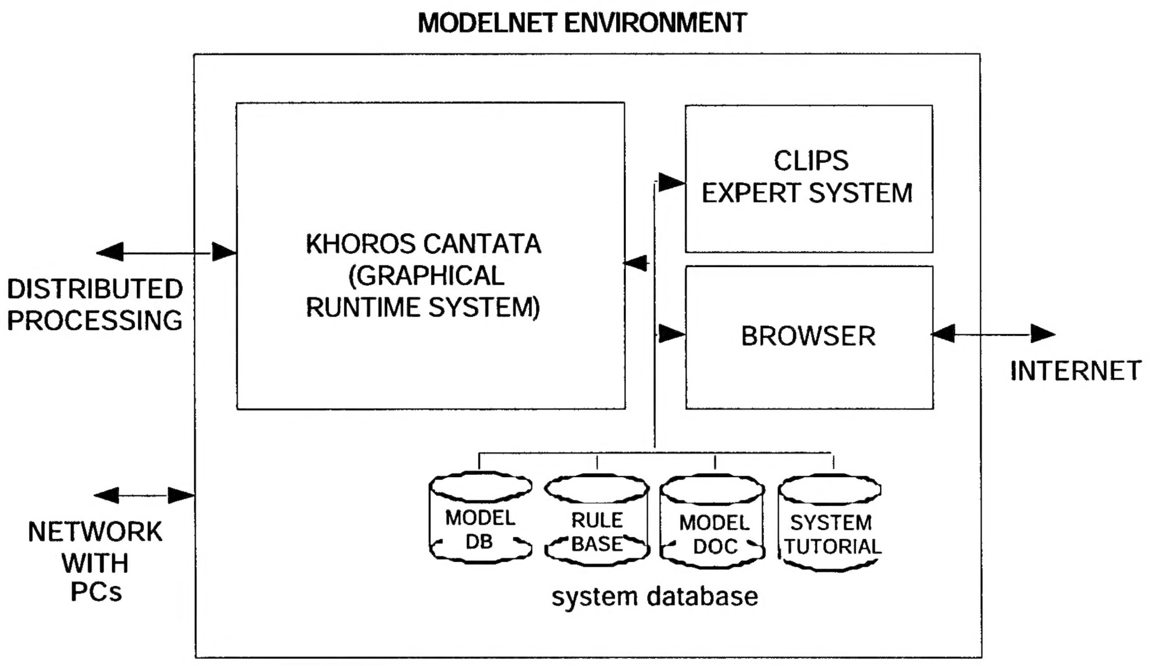


Figure2 - Example of the Khoros Cantata Visual Programming Interface for a Sensor simulation.

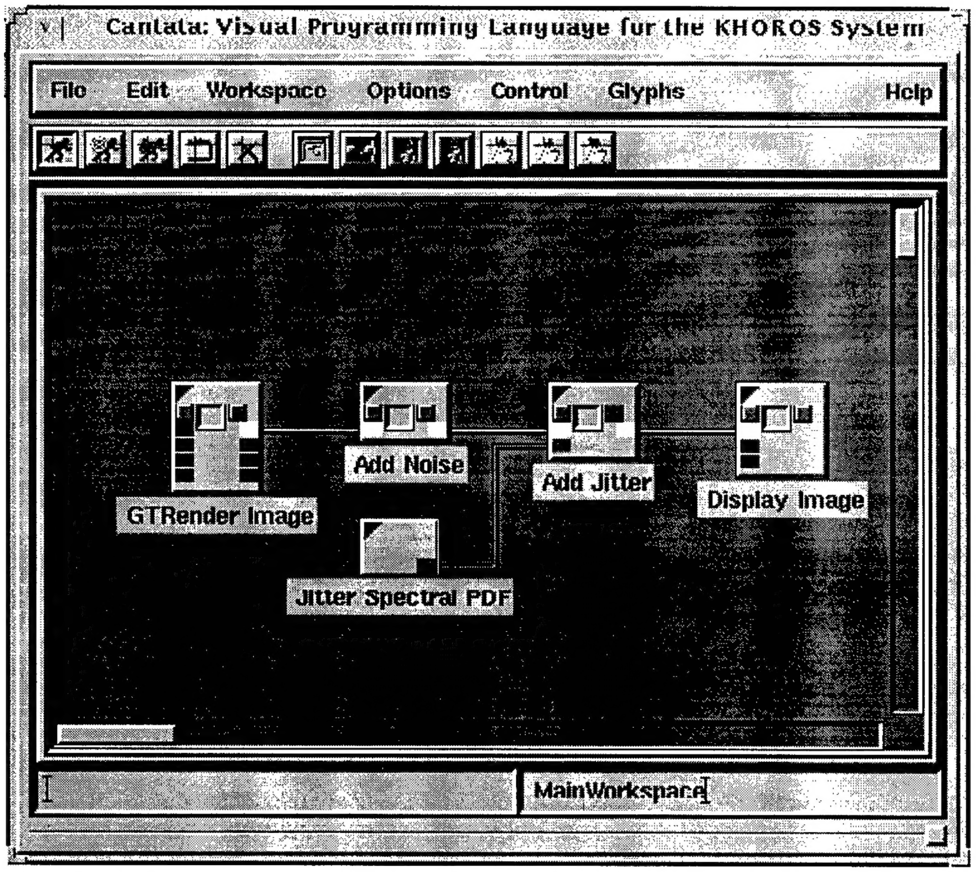


Figure 3 - Use of Khoros in the MODELNET environment.

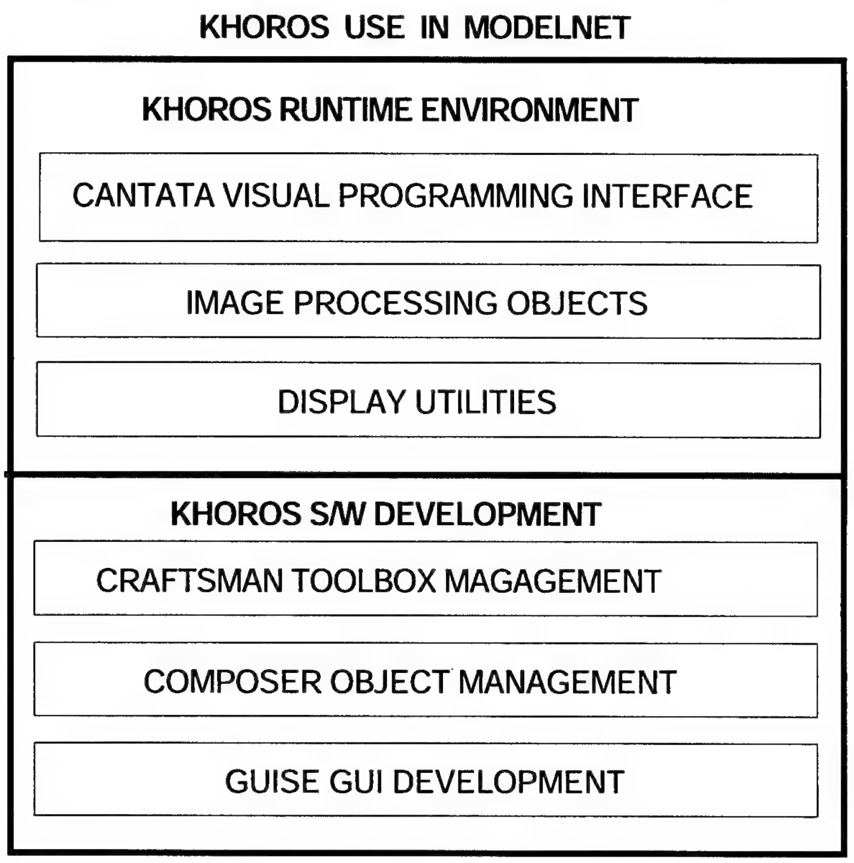

KHOROS USE IN MODELNET
KHOROS RUNTIME ENVIRONMENT
CANTATA VISUAL PROGRAMMING INTERFACE
IMAGE PROCESSING OBJECTS
DISPLAY UTILITIES
KHOROS S/W DEVELOPMENT
CRAFTSMAN TOOLBOX MAGAGEMENT
COMPOSER OBJECT MANAGEMENT
GUISE GUI DEVELOPMENT

Figure 4 - Use of Browser in MODELNET.

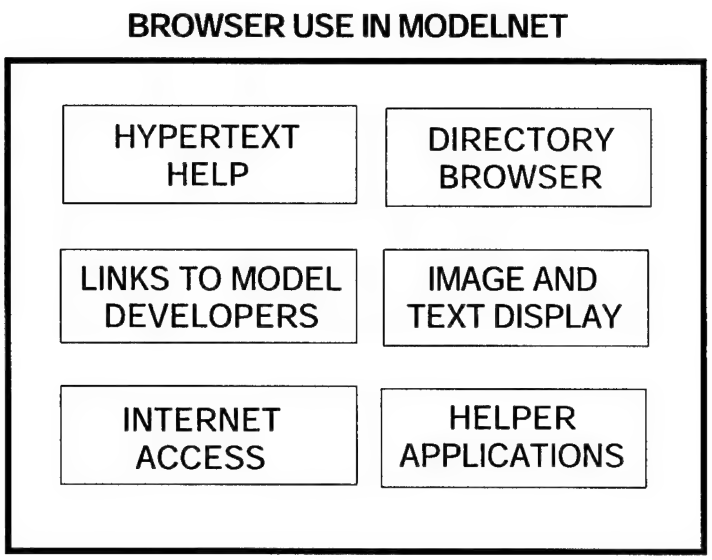
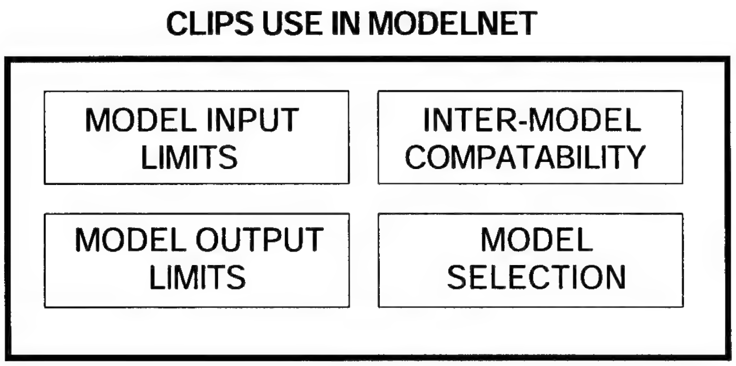


Figure 5 - Design of CLIPS use in MODELNET



# Implementing a Blackboard Based Information Fusion Architecture Using the CEENSS Methodology

Presented by:

Mr. Harold W. Dean
Mr. Paul W. Salchak
SYMVIONICS, Inc.
1312 Research Park Drive
Dayton, Ohio 45432

Track No:
SIT2

## Abstract

Two of the many problems confronting avionics developers today are the need to rapidly design, prototype, develop, and insert modern technology into avionics platforms and the need to support that technology under conditions where the pace of obsolescence is much smaller than the devices' required service life. SYMVIONICS' Complex System Information Fusion Tools (CSIFT) program is facing both of these issues. In this paper, we will present our findings and recommendations to-date as a case study of the goals, approach, and accomplishments of the program.

The CSIFT program uses a Data Activated Operating System (commonly referred to as a Distributed Blackboard), in the context of an offboard/onboard information fusion application, to implement a version of the Information Dispatcher function of the USAF's Airborne Information Fusion Architecture (AIFA). Our approach provides an efficient environment for prototyping and developing data oriented avionics applications.

The CSIFT program addresses the technology obsolescence issue by our use of the Continuous Electronics Enhancements Using Simulatable Specifications (CEENSS) methodology. This methodology provides for rigorous requirements modeling and formal verification, and promotes the use of vendor neutral, tool neutral simulatable specifications for electronic designs.

## Background

A multitude of problems confront modern military system designers. Few of the problems are new. There are always funding constraints. Force reduction cycles are common occurrences at periodic intervals. There are labor issues, requirements changes, scope changes, and others too numerous to mention. For the most part, these issues have been faced by system designers for centuries. The first of the great pyramids in Egypt was not finished as a true pyramid because halfway through the project it was discovered that the initial design could not support it's own weight. There was an OOPS!

There is one issue facing modern system designers that has no apparent historical equivalent: the exponential rate at which information and technology are increasing and changing. More accurately, we all have been on an exponential curve. Current system designs are becoming very close to the section of the curve that is going vertical. Twenty years ago systems were planned for fifteen to twenty year life cycles. Today, at the completion of a three-year design cycle, up to half of the parts in system may be obsolete and unavailable. Software is not immune to the problem. The processor systems increase in capability, allowing new features, or become obsolete, requiring re-engineering.

Without attempting to belabor the obvious, it is essential to provide systems that facilitate fast, responsive, efficient hardware and software development and economical hardware and software technology enhancements and reengineering. SYMVIONICS' Complex System Information Fusion Tools (CSIFT) project with the Sensor Directorate is developing a flexible software development environment, the Blackboard, and providing a test case for a developing methodology for reengineering support, CEENSS.

In order to understand both the Blackboard and CEENSS, we must provide some additional information on the CSIFT project. The information provided about the CSIFT project and Sensor Fusion in this paper is presented to provide context for the Blackboard development tools and the CEENSS methodology. The sensor fusion technology being developed on the CSIFT project is a model application, the Information Dispatcher, to be the front end for a Sensor Data Fusion suite.

## 2 Sensor Data Fusion

What is Sensor Data Fusion? First, a caveat on this explanation. This discussion is not intended to be the authoritative history of Data Fusion, but to provide context, primarily with an Air Force bias. In general, Sensor Data Fusion initiatives are under the overall umbrella of Situation Awareness research and development. In modern air combat, pilots have more sources of data and information than ever before in history. In the early days of air combat, the pilots had some instruments and their eyeballs. However, technology marched on. Today pilots have the most sophisticated aircraft ever produced and they have more information than any person should need to sort through. They have voice data and AWACS data and satellite data and real beam radar and Synthetic Aperture Radar and digital maps and IR data and camera data and and and _ If it wasn't so important to actually fly the aircraft, accomplish the mission, and stay alive, it might be possible for the pilot to spend all his time sorting through data to find just the right piece. Situation awareness research is focussed on providing tools to assist the pilot with his understanding of his highly dynamic environment. Sensor Data Fusion, in general, focuses on the aspects of situation awareness that relate to on-board and off-board sensor data collection, correlation, and display.

Although all services are performing situation awareness and sensor data fusion, the CSIFT program traces it's roots to the Joint Directors of Laboratories (JDL) Data Fusion Group. The Joint Directors of Laboratories (JDL) Data Fusion Group produced the Data Fusion Process Model shown in Figure 1 as a reference for continued information fusion refinement and evolution. Significant efforts continue to improve the architecture, processor, and methods associated with information fusion for on-board and off-board avionics systems.

Figure 1. Historical Reference: The JDL Data Fusion Process Model.

A major impetus behind data fusion is the increasing numbers and effective range of weapons and sensor systems, resulting in an information war. The current international political climate will provide opportunities forever more fusion of this new and improved sensor gathered information. This information war and new political climate affects the strategy and tactics in the employment of systems, making data fusion a key and complex process, pervasive throughout all C3I systems. While the impact of data fusion is large, it must be recognized that fusion technology is implemented as a part of sensor, weapons, and C2 systems, and not as an independent entity.

Data fusion is fundamentally a process designed to enhance the value of information for decision support. It is an essential, enabling process to organize, combine and interpret information from various sources which may contain numbers of targets, conflicting reports, cluttered backgrounds, degrees of error, deception, and ambiguities about events or behaviors. A high level representation of the data fusion process, illustrating its major elements, is provided in Figure 1. Level 1 processing, Object Refinement, combines parametric data from multiple sensors to determine the position, kinematics, attributes or identity of low level entities. Level 2 processing, Situation Refinement, develops a description or interpretation of the current relationships among objects and events in the context of the operational environment. The result of this processing is a determination or refinement of the battle/operational situations. Level 3 processing, Threat Refinement, provides estimates of enemy capabilities and enemy intent, identifies threat opportunities, and levels of danger. Level 4 processing, Process Refinement, monitors and evaluates the ongoing fusion process to refine the process itself, and to guide the acquisition of data to achieve optimal results. These functions interact with each of the data function levels and with external systems for the operators to accomplish their purpose.

Based on the JDL model, the Air Force Research Laboratory (formerly Wright Laboratories) continued an effort to develop a generalized, open, non-proprietary avionics information fusion architecture. This architecture, the Avionics Information Fusion Architecture (AIFA), depicted in Figure 2, provided an architectural standard which defined the different functional areas, the underlying components/techniques for each functional area, and the high level interfaces between each component. This reference has since been advanced through various contracted efforts and the work of the Open

Architecture Fusion Work Group (OAFWG). For our purposes here, though, and in the interest of a simpler example we will use the AIFA for our remaining discussion.

Figure 2. Avionics Information Fusion Architecture (AIFA).

The CSIFT project promotes concepts and solutions that will support the evolving fusion architecture standards and standard elements. Specifically, we are focusing directly on on-board/off-board information problem, the Information Dispatcher, utilizing a "Blackboard" approach. The Blackboard approach has the following characteristics:

Σ Architecture is Extensible
Σ Architecture in Scaleable
Σ Architecture accommodate multiple processes
Σ Additional processors are easy to integrate
Σ Architecture is data driven (explicit scheduling of algorithms is not required)
Σ Integration is "plug & play"
Σ No explicit communication required
Σ Architecture is hardware independent

## 3 Blackboard Paradigm

What is a "Blackboard"? Well they used to be big sheets of dark colored slate that you hung on the wall and wrote on with pieces of chalk. Then they became green "something " hung on the wall, but you still used chalk. Now they are white and you use dry erase markers. "Blackboarding" was also an early decision technique in Artificial Intelligence applications that used a software "panel of experts" to evaluate a problem, each expert suggested a solution, then the decision heuristic combined the results to obtain "the answer".

The CSIFT Blackboard is not exactly any of the above but it makes use of some of the concepts displayed in the use of the above "Blackboards". The CSIFT Blackboard is a software operating system "extension" that provides a structure for developing and executing data driven software applications. We have based our work and initial software implementation on a Data Activated Operating System (DADS), developed by the Northrop Grumman Corporation. Extensions to the paradigm are being developed via our CSIFT project, with Northrop Grumman as a subcontractor.

### 3.1 Blackboard Paradigm

What is the Blackboard paradigm? Envision a large room with a traditional blackboard on one wall. The room has desks with "experts", all of whom have a small section of the blackboard that they watch diligently. Figure 3 illustrates the concept. Every so often, someone comes into the room and puts a sticky note with information (data), on the blackboard. The "expert" responsible for that section grabs the data, takes it to his desk, performs his function, writes new data on another sticky note, puts it on the blackboard in another section. At that point either another "expert" becomes active or someone grabs the note and takes it out of the room. None of the "experts" really need to know what anyone else does with the data. There is no explicit coordination of effort. Nobody really has to "schedule" each task. If a better "expert" becomes available to take over a desk, the process is not disrupted (as long the data remains in the same format). In the software implementation of this paradigm we rename the "experts" as Functional Elements and we identify the data as Data Elements.

Figure 3. Generalized Blackboard Representation.

It must be stressed that this simple concept does extend to multiple, distributed blackboards for more complex applications. Our current solution does not restrict their number or physical allocation and placement in a computing environment. In fact, as has been shared, one of the most exciting aspects of this approach is its natural extensibility and affinity for larger distributed configurations!

## 3.2 Data Fusion Blackboard

Now we can consider an example of one Data Fusion architecture - the AIFA (simplified to provide a manageable example) is depicted in Figure 4. We began with the following assumptions: A blackboard architecture exists which allows transparent distributed access to data. There are two essential components, the data objects, which are untyped, named byte arrays, and the functional elements, which consist of executable code which reads the data objects, as well as activation logic which determines when the functional element code is executed based on logical conditions which may exist in the data objects to which the functional element subscribes. The blackboard may be supplemented by intelligent agents, which in this context are dynamically created functional elements which are retrieved and instantiated from some persistent database. Functional elements do not care where a particular data object resides. Sufficient networking resources and abstraction in the API are assumed to make this possible. The basic philosophy is that the data objects capture all state' information, and are accessible to any functional elements to be read. They are written by exactly one functional element. All procedures or operations are represented by the functional elements. We have established the candidate blackboard allocations (Figure 5) described in the following paragraphs.

Figure 4. Simplified AIFA Example Architecture.

Figure 5. AIFA Example Blackboard Allocation.

### 3.2.1 Information Dispatching

This is the functional process responsible for all interaction with the outside world of data. The Information Dispatcher receives all data, both from on-board and off-board sources, reconciles temporal discrepancies, and routes data into the fusion structure. The Information Dispatcher also handles the data being handed back to the outside world.

### 3.2.2 Object Refinement (Level 1 data fusion)

The basic question answered by level 1 data fusion is "who's out there?" The operations associated with level 1 fusion are hierarchically decomposed as detection, tracking and identification. This suggests a partitioning of a blackboard dedicated to level 1 fusion into three sub-blackboards that are assigned to these three areas.

### 3.2.3 Detection Sub-blackboard

Functional elements assigned to particular sensors process reports, perform the appropriate coordinate transformations and post data objects to the detection sub-blackboard. As new reports come in the functional elements are free to prune the data as desired. For example, an FE may post the last three reports as separate data objects, or it might post the most current, replacing it as necessary.

Tracking Sub-blackboard

This sub-blackboard may be further decomposed into two sub-sub-blackboards: data association and alignment and track maintenance. Functional elements associated with data association scan the detection sub-blackboard and decide which data elements might belong together. At this point, hard or soft decisions may be made. In a hard decision, new data objects are immediately output which represent several aggregated detection reports. In a soft decision, a matrix might be output which represents pairwise scoring. An optimization algorithm (e.g. Munkres) could be used by another functional element to solve for the best possible pairings. The data objects involved in the pairing could be the detection reports only, or the detection reports plus the ongoing tracks (assignment of reports to tracks for updating tracks). The track maintenance sub-blackboard, reads the output of the data association (data objects with assignments to tracks) and updates the track data objects, which are then posted on it's blackboard. With the track maintenance sub-blackboard are functional elements associated with estimation. These would be used in two ways: to produce a synthesis of measurements from various sensors (for example a radar capable of range and azimuth only with measurement from a height finder radar), or to time align data (this is necessary in track filtering) by propagating state information. The estimation functional elements would not only be used by the track filters, but by any functional element requiring their functions.

Identification Sub-blackboard

The functional elements associated with this blackboard implement model based detection. They scan the track sub-blackboard. Each functional element is looking for a specific pattern with it's activation logic. For example, an FE might be looking for tracks that might correspond to fighter jets, based on platform dynamics. When a track object fits the particular pattern searched for, the ID-FE declares a hit and posts a data object to the identified track sub-blackboard. As in data association, at this point either hard decisions or soft decisions (possibilities which have some score) may be made at this point. This sub-blackboard represents the output of the level 1 data fusion blackboard.

### 3.2.4 Situation Assessment (Level II data fusion)

This consists of three identically structured blackboards: one each for enemy force refinement, neutral force refinement and friendly force refinement. The basic questions answered at this stage are "what are the groupings?" and "what do they seem to be doing?" The basic paradigm used here is model-based detection. Each functional element scans all of the identified track objects at the output of level 1 fusion and looks for a particular grouping. As such, each embodies a hypothesis to be tested such as "there is a squadron of planes flying in the same direction at location xyz". The geo-force refinement logic could be implemented as a sub-blackboard whose FE's get to score and vote on associations produced in the situation assessment blackboard, based on geographical properties, These could be instantiated dynamically as agents which correspond to what is possible for a given geographic region. For example, a situation assessment FE, reading a group of tracks, triggers and outputs a data object corresponding to a hypothesized tank formation at location xyz. An agent responsible for location xyz notes that the location corresponds to the middle of a swamp and that heavy vehicles such as tanks are unlikely. It, therefore, does not post an output data object or assigns a very low score.

### 3.2.5 Threat Assessment (Level 3 data fusion)

The threat assessment or level 3 fusion blackboard takes inputs from the level 2 or situation assessment blackboard. These consist of data objects that describe the friendly, neutral, and hostile force structures and activities. The purpose of the functional elements used by the treat assessment blackboard is to reason about the possible outcomes of these activities, based on weapons characteristics, rules of engagement and orders of battle. Probably the most realistic way of mapping this into a blackboard architecture is to have a controlling functional element corresponding to the conflict wargaming module which spawns agents as necessary to play out scenarios. The weapons models, tactics, rules of engagement, and strategy would be resident in a long term database accessible to and used to structure the creation of agents who play roles in simulations. The wargaming module would pose situations, create the participant agents, play out scenarios, and analyze the outcomes. The output would represent the likely outcomes and would give an estimate of threat intent. One possible advantage to approaching this as a simulation (set of simulations) is that tactics and weapons used by friendly forces may be evaluated as well and improvements suggested. The blackboard would be partitioned into three parts, an input section, with annotated objects from situation assessment, a scratch pad" section, used by the conflict wargaming module used to run scenarios, and an output section with likely outcomes and intents. Since the conflict wargaming module can execute evaluations in parallel, it would be able to answer "what if?" type queries posed by functional elements elsewhere in the system. One possible concern is the computational resources used to run simulations. There are several possible ways to address this. First, the hardware cost of computing is declining extremely rapidly. Faster processors and new techniques such as reconfigurable logic are making this so. Second, simulations may be run at many levels of fidelity and detail. Several functional elements corresponding to the conflict wargaming module could exist, to run the simulations at the appropriate level, or to use either heuristic or analytical models to save time.

### 3.2.6 Planning Object

The purpose of the planning object is to make recommendations that improve situation awareness. The most logical way to implement this in a blackboard architecture is to spawn agents which get to snoop the operations on the level 1,2 and 3 blackboards and post their findings on a blackboard associated with planning. Functional elements would then read these data objects containing synopses of the other blackboards and generate plans. As in other places, methods based on model based recognition could be used to look for specific situations and propose a plan. The planning object is one place where considerable effort could be expended. The nature of the operations here implies a goal matching mechanism that seeks to optimize use of resources to satisfy a particular user need.

At the output of the planning blackboard, a final set of FE's would score these plans and output a final set which would meet some threshold criterion. One advantage of doing this is that the set would be presented as options with tradeoffs.

### 3.2.7 Dynamic Integrated Situation Representation

This seems very straightforward to map to a blackboard architecture. The output data objects of the planning, aircraft status, threat intent and situation assessment blackboards would be read by an FE (or hierarchical set of FE's) that would determine what to display and how to display it.

## 3.3 Key Attributes for the Blackboard Method - a User's Perspective

We have just discussed how a representative portion of the Avionics Information Fusion Architecture (AIFA) could be implemented as a Blackboard system. Due to the nature of the Blackboard, any expert processes can be substituted to create any fusion architecture. Given this ability, then, a critical aspect concerns the benefits to be gained from such an implementation, in the context of the user of the IF system. In this sense, the user can assume several roles: human or machine, actively interacting with or merely monitoring the IF system. Furthermore, these roles may be set in the context of different activities, such as designing or developing the IF system, upgrading it, maintaining it, and dynamically interacting with it under mission conditions.

The strength of the Blackboard lies in its ability to serve the user in all these roles and within all these activities. The primary benefit is derived from the consequential ability to provide the user with a consistent view of the system from cradle to grave, in all phases of design development, testing and use under mission conditions. This allows the IF system implemented in the blackboard paradigm to be totally user-driven.

### 3.3.1 What Blackboarding Achieves for an IF Architecture

The whole notion of Blackboarding is to provide a data-driven viewpoint of a processing system, without disrupting the effectiveness or efficiencies to be gained through a structured or Object-Oriented (O-O) implementation of the actual processing software. Blackboarding extracts the functional-flow perspective from within, for example, the hierarchical and inheritance-driven structure of an O-O design. By extracting this perspective, as illustrated in Figure 6, and by making the key components of such a viewpoint (functional elements and data objects) accessible. Blackboarding provides three critical benefits to an IF system, or to any system for that matter:

Σ Interfacing Flexibility and State Saving
Σ User's context Processing, and
Σ Robustness

Although these components already exist in the O-O implementation of the system software and communications structures, their extraction into the Blackboard paradigm provides these three additional benefits, which we now discuss individually.

ÆINVALID_FIELD: ObjectØ
Figure 6. A general Blackboard, indicating key components.

Interfacing Flexibility and State Saving - These two capabilities are illustrated in Figure 7. As shown, the state saving is accomplished by the functional elements and the interfacing flexibility is inherent in the data objects. Although the functions and their states certainly already exist within the system O-O implementation, by assimilating several functions to post data to a Blackboard, we capture the relational context of these functions, their states and the particular data as some functionality of interest to a user, which we will then make available to that user through an interface. In other words, this aggregation of state, function and data has relevance to a using system, intelligent agent, or human, so we make it available easily for interfacing and understanding of the relational context that matters, in a much more direct fashion than instrumenting the O-O structure to capture such information.
ÆINVALID_FIELD: ObjectØ
Figure 7. Capturing function, state, and data relational relevance in a Blackboard.

The myriad relational relevances of sets of function, state and data, are the very heart of an IF system. Hence, the ability to capture any number of these relational views is exactly one-third of the key to realizing a truly user-driven IF system under all the roles and activities we defined earlier. The second one-third has to do with maintaining the user's context, as we discuss next.

Processing in the User's Context - In an IF system, as well as many other complex systems, the user's view is not necessarily that of the software designer or system designer. It is, however, the user's view which should drive design, implementation or modification decisions. Obviously it is the user's view which applies to the use of the system.

Requiring the user to penetrate or understand the nuances of the software design or the computing system design is unreasonable. Especially in IF, where the fusion processing is complex in itself, we should not require the user to place their view of the functional flow and process sequencing in the framework of the software or system design. Whether the user is human or machine, active or passive, their view of the system should be the only view they need to deal with. The Blackboard technique provides such a view.

Consider the simplification of the blackboard view of some arbitrary functional elements and data objects of Figure 8. Here, we have overlayed the basic functional view of this particular system segment over the shadowed data object and function structure. This is precisely the view the user requires, with emphasis on functional flow, visibility of data, and access to functional state. This view is relevant to the user, without the encumbrance of the software or system representations. In the context of this view, the user may view, capture, or interface the IF system within the data-driven and functionally-concise view of importance.

Figure 8. Capturing the concise functional view through the Blackboard.

Now, having the ability to interface easily, maintain state, and view relevant portions of the IF system in the user's context, we have two-thirds of what we need for an effective, user-driven IF system under all the roles and activities we defined earlier. The final one-third has to do with robustness, as we discuss next.

Robustness - In the IF context, robustness has to do with improving design or development effectiveness, easing future modifications and upgrades, and custom-tailoring the IF system to a particular mission need. In all of these areas, the IF system needs to permit the addition, deletion, or re-organization of functional elements and the data objects relevant to the activity in a simple and intuitive manner. As illustrated in Figure 9, the Blackboard system permits such ease of change.

ÆINVALID_FIELD: ObjectØ

Figure 9. Robustness of the Blackboard system.

Here, we note that adding (or deleting) functional elements, rearranging connectivity, or posting additional data objects to the Blackboard may all be done easily and in the user's context. Whether for design, testing, operational optimization, or any other reason, the changes can be done purely in the user's functional context. Hence, the user's needs, whether for human or machine interface, interactivity, or simple monitoring, are always clear in the view of the system which is to satisfy these needs.

In summary of the Blackboard's ability to support the general task of IF system implementation and application, the ease of interface, state saving, view in the user's context, and robustness make this technique especially beneficial. Fundamentally, Blackboarding extracts the appropriate view or views, whatever they may be, for the user, whatever or whoever that may be, in the user's particular activity. As Figure 10 shows, it extracts the concise, relevant process view from the "clutter" of the (typically) O-O software system executing upon some processing system or systems. Optimal software system design is not being trivialized here, and the benefits to the overall quality of a software design from O-O's hierarchical, modular, inheritance-driven paradigm are valid and necessary. However, these issues are the domain of the software designer, not the IF user. The Blackboarding technique permits both systems to co-exist efficiently, and provides a means for the user to extract the proper view of the system functions and their important relational aspects.

ÆINVALID_FIELD: ObjectØ

Figure 10. O-O and Blackboarding - an effective separation of view.

## 3.4 Blackboarding as a Design Aid

Consider the design representations of Figure 11. Here, we see the division of viewpoints for the different design disciplines. The Application Designer, in the IF situation, is mission-conscious and process-oriented. This designer seeks to organize some arrangement of processes to refine source information such that information critical to the mission is most effectively handled in a decision support role. The Application Designer is conscious of the need for efficient and cost-effective software and system design, but is usually more in the role of setting requirements for such endeavors than actually considering them. The Software Designer lives in the world of modularity, reusability, and O-O design methods by which these, and other cost-effectiveness goals,

may be accomplished. The process and data flow of the Application Designer exists in this realm, but the Software Designer concentrates more upon vertically integrating processes, through nested methods and inheritance aspects under the O-O paradigm. The System Designer works in the realm of databusses, processors, and physical constraints. This designer must support the requirements of the Software Designer through effective organization of processing and storage elements and their communications links. The process and data flow of the Application Designer is remote to this area, where the requirements from the Software Designer drive the system implementation.
ÆINVALID_FIELD: ObjectØ
Figure 11. Multiple perspectives of the design process.

Since the Application Designer needs an understanding of the process and data flow as the design is producing, but not necessarily the O-O structure of system configuration, the Blackboard provides an ideal means to separate and present that view. The Blackboarding can capture the relevant processes and data objects to an aspect of interest to the Application Designer, who can view this process and data flow perspective as the design matures. All the Application Designer requires is a concurrent development of the Blackboarding elements (Blackboard, interfaces) with the software and system development. Since software development is generally conducted in a spiral fashion, with iterative and progressively complete implementations at all levels of the design, concurrent implementations of the Blackboard and its interfaces will keep the Application Designer focused precisely upon the relevant aspects of the design to satisfying the IF requirements.

From this example, we can draw some conclusions. The Blackboard interfacing must do only one thing: configure the design data for its intended application as a design visualization tool. Unlike the dynamic application situation, there is no need for any kind of "state" information, as the state of the design IS the design. We can extrapolate this simple example to the other aspects of life-cycle as well. If the Blackboard can interface to the supplier system in any of its manifestations, design, prototype, multi-iteration versions, pre-production, and final, then the Blackboarding system can be the Application Designer's tool universally, from cradle to grave. In the test manifestations, state-saving once again becomes relevant. Basically, we can use the Blackboarding technique as the designer's viewer, subject to the same two requirements we defined in testing the flexible interface and state-saving hypothesis:

Σ The input and output interfaces to and from the blackboard must be possible, and
Σ State information must be provided along with control or information data.

We can conclude from this simple example and its extrapolation to the more complete life-cycle, that our hypothesis regarding the ability to provide a consistent user's view of the AIFA is valid, and that such a view has value to all aspects of an IF system's life-cycle, subject to the requirements we asserted. Again we defer more comprehensive discussions of the requirements to the conclusion of this section, and continue into the application aid case.

## 3.5 Blackboarding as an Application Aid

To optimize some facet or facets of an IF engine, visualization or AI-processing systems (intelligent agents) need to capture process and data information precisely into their particular user's view of the system and data. Our studies revealed that the user, which could be an intelligent agent (machine or human), could test hypotheses based upon the data presented through the blackboard, to optimize the IF system via control functions. For example, assume a mission manager function in the AIFA architecture, which includes an intelligent agent whose purpose is to optimize the probability of optimal sensor mixing.

Here, similar to the case we considered under the flexible interfacing and state-saving proof, our Blackboard includes a posting interface and some output interfaces. The Blackboard captures several information and control data objects, including:

Σ The Resource Manager commands to the Information Sources which reveals how these resources are being commanded,

Σ The information output of these sources, which reveals the results of executing the commands,

Σ The information provided to the Enemy Force Refinement from the Information Dispatching, which reveals the information upon which it is acting, and the result of the Threat or Target Intent processing, which is the conclusion of this processing sequence.

In other words, the Blackboard captures what we are commanding, what information we are getting from the sources, and what our intent processing is deciding based upon such information. We may then provide some or all of this information, with its associated state, in whatever format we choose to the human user, by interfacing it to the PVI interface function. We may also provide this information, via a system interface, to some intelligent agent operating under the Mission Manager to optimize this sub-processing sequence within the AIFA. Hence, we have the ability to form the user's view from the AIFA, in a particular context for a particular reason, and could easily do so for myriad other context and reasons, subject to the same requirements we have seen for the previous tests:

Σ The input and output interfaces to and from the Blackboard must be possible, and

Σ State information must be provided along with control or information data.

We can conclude from this example and its extrapolation to the more general case, that our hypothesis regarding the ability to provide a user's view as a dynamic application aid is valid, subject to the requirements we asserted. An once again we defer our discussions of the specifics of these requirements to the conclusion of this section. Next, we look into the robustness issues.

## 3.6 Implementing Future Modifications

The Blackboard approach is essentially "made to be modified." As illustrated in Figure 12, upgraded or new processing entities can be integrated with a process and function view (application designer), as well as the necessary software and system implementation views (software/system designers). Any new processes and their associated data appear in the process and data flow view to the Application Designer, as methods and data in the inheritance of the O-O structure to the Software Designer, and as processing, storage, and communications loads to the system designer.

Another aspect of robustness which our investigations revealed is that the consistency in user's view provided by Blackboarding can not only extract the proper perspective in the design stages, but can actually enhance the whole process. The Blackboard perspective remains consistent across all portions of a software designer's iterative development paradigm: design, implementation, testing, assessment, revision, and so forth. This consistency of view ensures that the Application designer is responding to issues raised in the various iterative development stages with the proper understanding.

ÆINVALID_FIELD: ObjectØ

Figure 12. Upgrading with the multiple-perspective views.

### 3.6.1 Providing Customized, Interactive User Interfaces

This capability can build directly on top of the blackboard context, again leaving the specific implementation of the software and hardware system functionality to their respective domain specialists. We basically proved this capability in the context of the previously discussed Intelligent Agent addition to the AIFA. The only additional issue is if we wish to implement such customized interfaces in-mission. Then we must determine whether the user's view of the system can actually be reconfigured in a manner that it does not disrupt ongoing IF functions critical to mission performance. The issue is not whether or not it can be done, but how to synchronize such an action into the dynamic mission execution. Hence, we see an additional requirement: the customized interfaces, if they are to be dynamically reconfigured in-mission, must not disrupt ongoing IF or other mission processing functions.

Basically, ignoring the in-mission aspects, the ability to customize the IF system for the user through the Blackboarding technique can be thought of as a layered system, with each layer in a particular specialization domain, and the focus of the whole implementation is upon consumer (user)-driven IF system optimization to satisfy diverse and changing mission needs. This is illustrated in Figure 13. As we disclosed earlier, the entire blackboard concept is also made to be modified, since the same consumer-driven paradigm for application also holds true for design, implementation and modification. By providing the user a consistent view of the system in all phases - design, implementation, modification, and application, we ensure a complete, user-driven system. Since it is an open architecture, data-driven, and object-oriented, ease of enhancement is built-in right from the beginning.

ÆINVALID_FIELD: ObjectØ

Figure 13. A layered perspective of the Blackboard as applied to IF.

## 3.7 Blackboarding and an Information Fusion Testbed

We have discussed three principal features of Blackboarding which demonstrate its support within an Information Fusion architecture. These same three attributes also permit the Blackboarding technique to serve as a common integration framework for the purpose of implementing a Distributed IF Testbed - which can serve all developers, testers, and users of IF technology in their particular missions and endeavors. In this section, we describe how this support for an Information Fusion Testbed can be achieved.

We begin with an abstraction of the view of Blackboarding for IF we presented at the conclusion of the previous section. Basically, this is a layered view of the Blackboard and its interfaces implemented upon some set of applications executing upon some array of processing systems. The interfaces basically provide users, or consumers, in-context access to the application/system sets, or suppliers. If we wish to make such a structure compatible with many different consumers or suppliers, we need only to implement the appropriate interfaces, as illustrated in Figure 14.
ÆINVALID_FIELD: ObjectØ
Figure 14. Multiple consumer/supplier interaction via the Blackboard.

The general strategy for the evolution and optimization of IF technology is to continue developing complementary Air-to-Air (A-A) and Air-to Ground (A-G) IF technology, by implementing on the ground first, and then transitioning these technologies to the air. An critical aspect of this paradigm is that it requires the development of a testbed, that is, an infrastructure and technology base to permit the development and testing with maximal use of real-world data and cooperation and collaboration of world-class talent. A distributed IF Testbed is the framework by which such talent, data, and technology may be organized collaboratively to develop the IF we need for the next century. The Blackboarding technique can provide that framework, through its ability to interface myriad applications, systems, and users, and to provide these users their necessary viewpoint and context. We will demonstrate this concept by beginning with a snapshot of a real-time embedded IF architecture, set in the Blackboarding framework. Then we will show how this system's development, testing, and implementation may be achieved under a Blackboard-supported testbed concept.

Consider a Fusion Manager function, as illustrated in Figure 15. Typically, and consistent with the AIFA or JDL fusion paradigms, such a function would be engaged in gathering information from source objects such as sensors and CNI systems, and invoking stages of object and threat refinement to provide a threat or target assessment to the Mission Manager. In this subsystem of the overall IF process, the consumer is the Fusion Manager, which receives information in a Fusion Manager's view of the situation, through an interface to the Blackboard. Information from the sources each have their own interfaces to post data (information and state) to the Blackboard. in an active mission, the Fusion Manager would also direct a sensor manager to reconfigure the source (sensor) mix and modes, although we do not show that aspect in the figure.
ÆINVALID_FIELD: ObjectØ
Figure 15. Blackboard supporting a Fusion Manager - real-time application.

Now, consider how this Fusion Manager might have been developed and tested, using Avionics Laboratory Assets. Assume that the MBV (Model-Based Vision) Laboratory had some sensor models which would serve well as test benches for the sensors, and that JMASS included environment and CNI models which would provide realistic simulation of these aspects. An IF Testbed would need to integrate these two resource laboratories (MBV and JMASS), and maintain the Fusion Managers view, to permit development and testing.

Figure 16 shows how the Blackboarding can support this activity. Here, the Blackboard provides the means to permit the KHOROS-framed MBV Lab sensor models and tools to interact with the JMASS-framed environment and CNI models. The Blackboard also extracts the Fusion Manager's viewpoint, and can support multiple subscribers (consumers) in a situation where different individuals or organizations may be responsible for, or interested in, different aspects of the Fusion Manager design, development or testing. The MBV Lab is not concerned about the fact that they operate in a KHOROS framework upon Unix systems, and that JMASS operates in a Solaris environment. The JMASS facility is similarly unconcerned about inter Lab particulars. The developers, designers, or testers (subscribers) viewing the simulation results are unconcerned with any of these inter-lab application or system issues, and concentrate on the function and flow in the context of their needs. The Blackboard enables all of this.
ÆINVALID_FIELD: ObjectØ
Figure 16. Blackboarding managing lab assets for subscribers in a development or testing application.

Now we can extend this concept to a distributed situation. Let us now assume that we are using not only Avionics Laboratory assets, but contractor's or other government facility assets at geographically-dispersed facilities. We might also consider that the development or testing is a collaborative effort, involving users (consumers) at geographically-dispersed locations. Implementation of a Distributed IF Testbed, under the Blackboarding paradigm, is again merely a matter of proper interfacing, as shown in Figure 17.

In this situation our interfacing has two classes - local and remote. Here, a CORBA (Common Object Request Broker Access) implementation would serve well as the local interfacing paradigm. Our figure is of course a simplified view, as CORBA shells and other CORBA structures would permeate into the applications in a fashion more than a simple interface "block." A Java-based remote interfacing construct would serve the remote users very well, as the Java applet and byte coding capabilities would provide browser-based consumer-supplier interaction as well as visualization of the testbed operation and execution. In fact, a Java-based, "Intranet" implementation, using CORBA at the intermediate levels, might be a good, common access paradigm for the IF Testbed in general.
ÆINVALID_FIELD: ObjectØ
Figure 17. Blackboarding to enable local and remote access to a geographically-distributed IF Testbed.

## 4 A Word About CEENSS

The Continuous Electronics ENhancements using Simulatable Specifications (CEENSS) program seeks to establish methods and tools for stable, repeatable, manufacturable electronics designs by focusing on formal requirements modeling, requirements verification through mathematical proof, and a complete, unambiguous, executable specification - called the SimSpec. The overall concept, depicted in Figure 18A, has shown outstanding improvements in overall design and design maintenance time - especially over a life-cycle of change, reducing risk and cost-to-market. A composition of a SimSpec is shown in Figure 18B. The CEENSS approach also centers on the use of 'Design Shelving' as a critical aspect of its Product Development Process. This framework is summarized in Figure 18C.

Figure 18. The CEENSS Concept.

Figure 18B. The CEENSS SimSpec.

Figure 18C. The Design Shelving Framework.

SYMVIONICS has successfully applied the CEENSS Spiral Development Methodology and many of the projects' tools during the CSIFT program. These "lessons learned" have been shared through involvement in the CEENSS Industry Review Board (IRB). For more information about the CEENSS program, and its current status and progress, refer to the web-site at http://www.ececs.uc.edu/~kbse/ceenss/.

## 5 Summary

In conclusion, the CSIFT Blackboard represents an exciting step in high performance distributed operating environments, with potential applications well beyond its information fusion roots. We have seen outstanding characteristics in both development (rapid, expressive, extensible, partitionable) and execution (behavior, performance) references. Our work will result in a set of open architecture tools and hardware demonstrating the initial off-board/on-board fusion information dispatcher, but also suitable for experimentation and application to other domains. We welcome any opportunity for discussion, critical analysis, additional research, or transition into developmental applications. Contact either author at (937) 426-4504, or through psalchak@symvionics.com.

Biographies:

Mr. Harold W. Dean

Mr. Dean is currently the Director of Programs for Research & Technology for SYMVIONICS, Inc. He is a graduate of The Ohio State University with a Bachelors degree in Electrical Engineering. He has 20 years experience in the design of military electronics systems. His primary expertise is embedded processors, aircrew training simulators, and high speed data acquisition and distribution systems. He is the co-inventor of the "Method for Simulating High Resolution Synthetic Aperture Radar Imagery From High Altitude Photographs," Patent Number 5,353,030 issued 4 October 1994. He has previously published the paper "Network Switching in the Virtual Test Station (VTS)" which was presented at the Test Facility Working Group Conference (TFWGCON), 1995.

Paul W. Salchak

Mr. Salchak is currently sharing responsibilities as a Staff Scientist and Director of New Products & Programs for SYMVIONICS, Inc. He is a graduate of Rochester Institute of Technology with a Bachelors degree in Electrical Engineering, and has completed a variety of graduate and specialized studies nation-wide. With over 19 years of experience in high-performance human interactive systems, Mr. Salchak is an expert in system analysis and development, electronic systems design automation, simulation-based development methods and tools, processes and metrics, training systems, simulation technologies, high-performance computing environments, and avionics systems, sensors, and displays. He is the inventor of the rapid-concept-to-market technique, Assertive Development, used by SYMVIONICS' research programs to define, insert, qualify, and transition advanced technologies. Recent research topics and reports have spanned such diverse areas as information distribution and leveling, chaotic signal processes and discrimination, collaborative/distributed decision support techniques, hardware/software co-design, Diminishing Manufacturing Sources (DMS), and Distributed Mission Training (DMT). Mr. Salchak's most recent publications have included "Supporting Hardware Trade Analysis and Cost Estimation Using Design Complexity" (presented at the VIUF National Convention/Fall '97) and "Innovative ATR Through Statistical Signal Processing and Chaos" (presented at the recent GOMAC '98).

## PLEASE CHECK THE APPROPRIATE BLOCK BELOW:

-AQ # ____________

☐ __________ copies are being forwarded. Indicate whether Statement A, B, C, D, E, F, or X applies.

☒ DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED

☐ DISTRIBUTION STATEMENT B:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES ONLY; (Indicate Reason and Date). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT C:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND THEIR CONTRACTORS; (Indicate Reason and Date). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT D:
DISTRIBUTION AUTHORIZED TO DoD AND U.S. DoD CONTRACTORS ONLY; (Indicate Reason and Date). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT E:
DISTRIBUTION AUTHORIZED TO DoD COMPONENTS ONLY; (Indicate Reason and Date). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT F:
FURTHER DISSEMINATION ONLY AS DIRECTED BY (Indicate Controlling DoD Office and Date) or HIGHER DoD AUTHORITY.

☐ DISTRIBUTION STATEMENT X:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND PRIVATE INDIVIDUALS OR ENTERPRISES ELIGIBLE TO OBTAIN EXPORT-CONTROLLED TECHNICAL DATA IN ACCORDANCE WITH DoD DIRECTIVE 5230.25, WITHHOLDING OF UNCLASSIFIED TECHNICAL DATA FROM PUBLIC DISCLOSURE, 6 Nov 1984 (Indicate date of determination). CONTROLLING DoD OFFICE IS (Indicate Controlling DoD Office).

☐ This document was previously forwarded to DTIC on __________ (date) and the AD number is ______________.

☐ In accordance with provisions of DoD instructions, the document requested is not supplied because:

☐ It will be published at a later date. (Enter approximate date, if known).

☐ Other. (Give Reason)

**DoD Directive 5230.24, "Distribution Statements on Technical Documents," 18 Mar 87, contains seven distribution statements, as described briefly above. Technical Documents must be assigned distribution statements.**

Tom Minnich
[illegible]
per phone call with J. Camp 10/22/98
**Authorized Signature/Date**

AFRL/FSD
2241 Avionic Circle
Wright -Patterson AFB, OH 45433-7318
John Camp
**Print or Type Name**

937 255 2164 3562
**Telephone Number**